MODELING SPACE SYSTEMS WITH THE DATA SYSTEMS DYNAMIC SIMULATOR PLUS

Christopher Rouff
NASA Goddard Space Flight Center
Code 522.1
Greenbelt, MD 20771
(301) 286 2938
chris.rouff@gsfc.nasa.gov

Philip Message
Stanford Telecommunications Inc.
7501 Forbes Boulevard
Seabrook, MD 20706
(301) 464 8900
pmessage@stel.gsfc.nasa.gov

ABSTRACT

The Data Systems Dynamic Simulator Plus (DSDS+) is a discrete-event-based simulator used to simulate complex, high-data-rate, end-to-end systems. It was developed for the specific purpose of evaluating candidate data systems for spacecraft. DSDS+ features a graphical user interface and hierarchical modeling. A data system is modeled in DSDS+ by selecting pre-programmed elements from the simulation library, placing them in a work area, and connecting them together using direct manipulation to form a model of an end-to-end system. When using DSDS+ the user has access to a library of standard pre-programmed elements. These elements represent tailorable components of NASA data systems and can be connected in any logical manner. Additional elements can also be developed which allows the more sophisticated user the option of extending the standard element set. Next, DSDS+ supports the use of data streams simulation. Data streams is the name given to a technique that ignores packet boundaries, but is sensitive to rate changes. Because rate changes are rare compared to packet arrivals in a typical NASA data system, data stream simulations require a fraction of the CPU run time.

1. INTRODUCTION

The Data Systems Dynamic Simulator Plus (DSDS+) is a general-purpose, discrete-event-based simulation tool. DSDS+ features a graphical user interface and hierarchical modeling. A data system is modeled in DSDS+ by selecting pre-programmed elements from the simulation library, placing them in a work area, and connecting them together using direct manipulation to form a model of the end-to-end system. Examples of simulation elements are data generators, CPUs, switches, as well as orbit calculators, schedulers and many others. If desired, users can also add their own modeling elements to the library to extend the base classes supported.

Development of the Data Systems Dynamic Simulator (DSDS) started in the late 1970's at Marshall Space Flight Center. Under contract to NASA, the General Electric Company was tasked to build a discrete event simulation tool especially suited for modeling NASA end-to-end data systems of the Space Shuttle and Space Station eras. Since then, DSDS has been in continual use. In 1985 the management and control of DSDS was transferred to Goddard Space Flight Center (GSFC).

Since management of DSDS has moved to GSFC, several changes have been made. First, an optimization technique called data streams was added to the simulator. Second, the character based user interface was changed to a graphical user interface, with a name change to DSDS+. In addition to the above, many simulation elements have been added, modified and streamlined. Stanford Telecommunications has been the primary contractor for the above modifications.

2. DSDS+ OVERVIEW

Models are developed pictorially in DSDS+, using a graphical user interface that provides close correlation between the model representation and the real system [1]. The user interface was developed on the X Windowing System [2] using the Interviews toolkit [3]. It allows models to be developed in a What You See Is What You Get (WYSIWYG) manner. Simulation elements are placed on the screen and connected together through direct manipulation. Users interactively select elements they want to use, click on the screen where they would like them to be placed, interactively set parameters of the elements, and connect elements together by clicking on corresponding input and output pins.

DSDS+ also supports hierarchical modeling, to any depth required, so that complex models can be decomposed into a series of detailed sub-level models. Users are also allowed to export parameters which are not set until the simulation is actually run. This allows simulation parameters to be changed between simulations without having to modify elements each simulation run.

In a DSDS+ model it is assumed that data is sent between elements in a packet. The packets can be simulated in one of two modes: packet by packet or as a data stream. Using packet simulation, packets are sent through the system one at a time. The data stream methodology simulates the changes in rate at each instant in time (rather than the individual packets constituting the rate change), and hence the speed at which the simulation runs is independent of the number of packets flowing through the system. NASA's space-based and ground-based data processing and communications systems are required to process extremely high data rates and data volumes, and simulating these systems using the traditional packet methodology would take months or years of CPU processing time - however, the data stream methodology enables these simulations to complete within just a few minutes. Both simulation techniques are discussed further in a later section.

DSDS+ contains an extensive library of pre-programmed simulation elements [4]. Examples of the pre-programmed elements include: data generators and sinks, data processors (e.g. CPUs with various service disciplines), buffers and queues, and data switches and routers. Each of these elements simulates a particular function or service, which may be tailored by the user to represent the specific application being modeled. For example, the data generator has a list of parameters associated with it that enable the user to define characteristics such as the packet sizes to be generated, their inter-arrival times, their priorities, etc. If desired, multiple instances of an element may be included in the model (e.g. multiple data generators), and each instance will have its own set of parameters defining the specific operations being simulated.

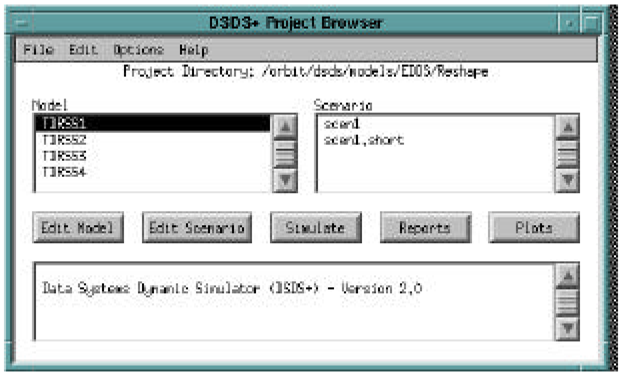
From the results of the simulation the sizes of buffers, the speed of processors and the speeds of data links that will be needed for the system to have a particular performance can be determined. Though DSDS+ itself does not process cost information, once the types of components are determined, the modeler can then determine the price of the proposed system.

3. CREATING MODELS IN DSDS+

The following discusses the user interface of DSDS+ and how a modeler would go about developing and executing a model. It discusses the main window (project browser), the model editor, the scenario editor, the simulator, the report browser and plots window.

Project Browser

The main DSDS+ window is the Project Browser (Figure 1). It provides access to the various editors and browsers that comprise the DSDS+ system. The current project directory is shown just beneath the Project Browser menu bar. The models in the directory are shown in the left-hand list, and the scenarios pertaining to the selected model are shown in the right-hand list.



Project Directory: /orbit/dsds/models/EDOS/Reshape

TDRSS1
TDRSS2
TDRSS3
TDRSS4

scen1
scen1,short

Data Systems Dynamic Simulator (DSDS+) - Version 2.0

Figure 1: Project Browser window

The "Edit Model" button opens the editor where simulation models are defined. The "Edit Scenarios" button opens the editor where a model can have multiple scenarios associated with it, each defining different run-time conditions for the model (e.g. different random number seeds to be used). The "Simulate" button runs the simulator on the selected model and scenario. The "Reports" button displays numeric results of the simulation and the "Plots" button allows the user to view graphs of collected data.

Model Editor

The Model Editor along with the Tool Box and a sub-model is shown in Figure 2. The model editor enables the modeler to create and edit hierarchical models. The main features of the editor are the menu bar across the top, the status bar below the menu bar, the work space in the middle, and the scroll bars and zoomers on the bottom and right edges.

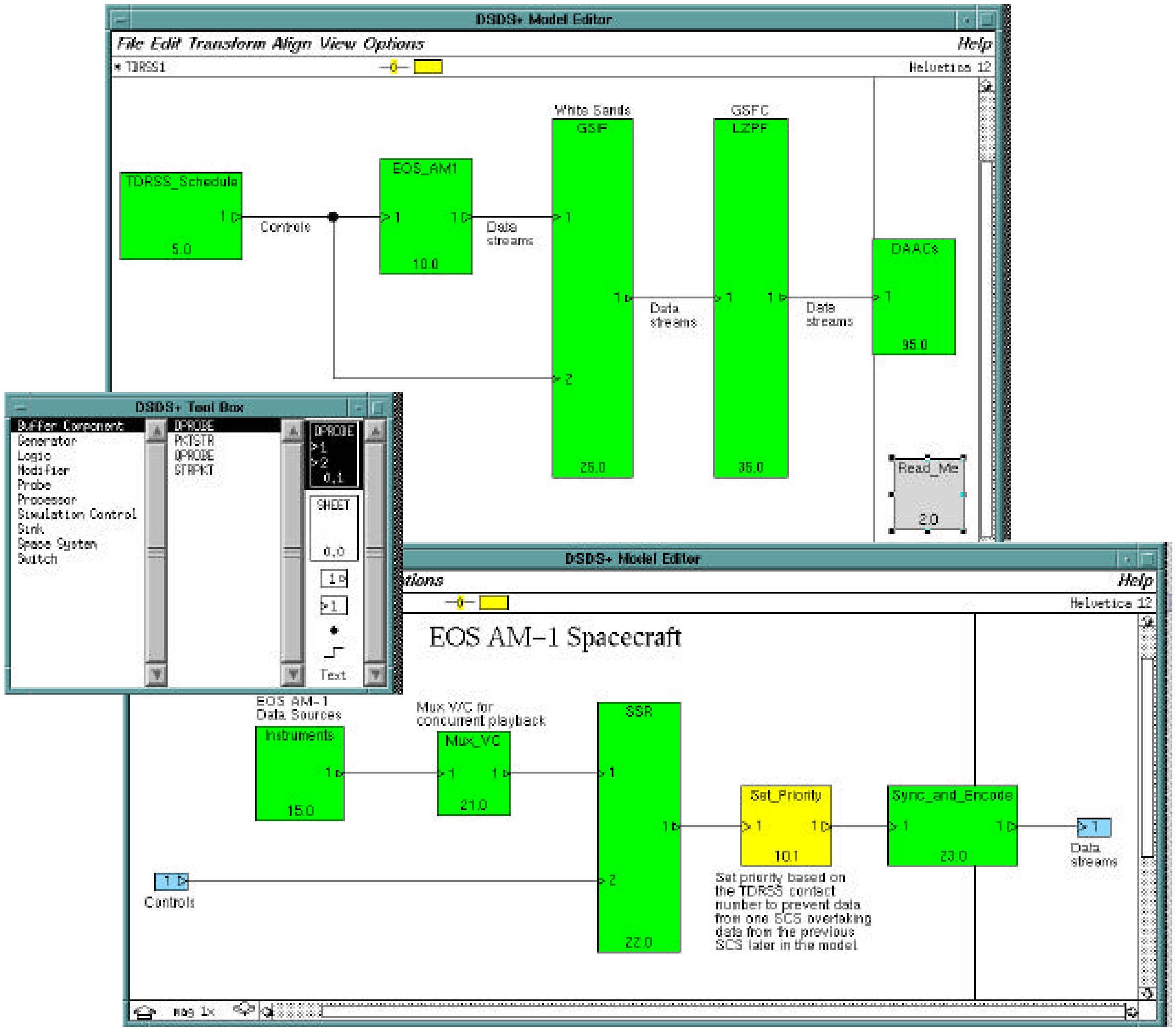
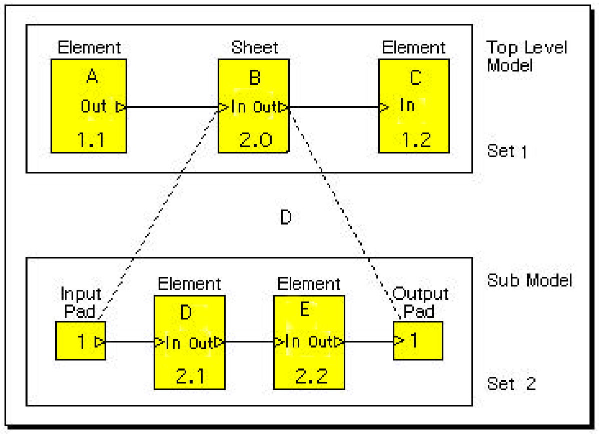


Figure 2: DSDS+ Model editor, Toolbox and subsheet.

The Toolbox, which appears whenever there is at least one Model Editor open, is used to select the type of object to be added to a model. This type is the highlighted object shown in the rightmost column. The objects that can be used in a model are simulation elements, sheets (to indicate a sublevel for hierarchical models), off sheet connectors, junctions for fan-in or fan-out of links, links, and free text for labels. Elements are divided into classes. By selecting a class in the left hand column of the toolbox, the corresponding elements that belong to that class appear in the middle column. When an element is selected, it's graphical representation is shown at the top of the rightmost column.

Connections are indicated by a sequence of contiguous horizontal and vertical line segments between an input and an output pin, or between a junction and an input or output. The vertical line on the right hand side of both model editor windows in Figure 2 indicates where a page break would appear when printed. Selecting the link object in the Toolbox's palette enables the modeler to override the automatic routing and perform manual routing. Note that once a link is formed, either automatically or manually, it will be automatically rerouted if either of its ends is moved.

Hierarchical models may be built by placing a composite "sheet" element in the workspace; the modeler can then "enter" into this sheet, and add elements on it, or even add lower-level sheets. The hierarchical support allows models to be developed in a structured manner which reduces the complexity of large models. In Figure 2, the model in the lower window is an example of a submodel which defines the “EOS_AM1” element in the upper window. Figure 3 illustrates this concept.



F

igure 3: Illustration of a hierarchical model.

Each element in a model simulates a particular function or service, tailored by a set of user-supplied values to represent the specific characteristics of the system being modeled. For example, the user-supplied parameters for the Source element (shown in the element editor in Figure 4) include the number of transactions to be generated, the transaction size, their inter-arrival times, etc. If desired, multiple instances of an element may be included in a model (e.g. multiple data Sources), and each instance will have its own set of parameters defining the specific operations being modeled.

Scenario Editor

Scenarios are a set of simulation program arguments that control how a model is executed. The Scenario Editor is where compiled models are setup for various runs. Each run can have a different duration and different random number seeds and can collect different statistics without changing the model. Each model can have multiple scenarios associated with it, each defining different run-time conditions for the model (e.g. different random number seeds to be used). By separating scenarios from models, it is possible to perform multiple runs for a given model to compare results without having to have separate models. The Scenario Editor is identical to the Element Editor in operation, except that the parameters apply to a scenario instead of an element.

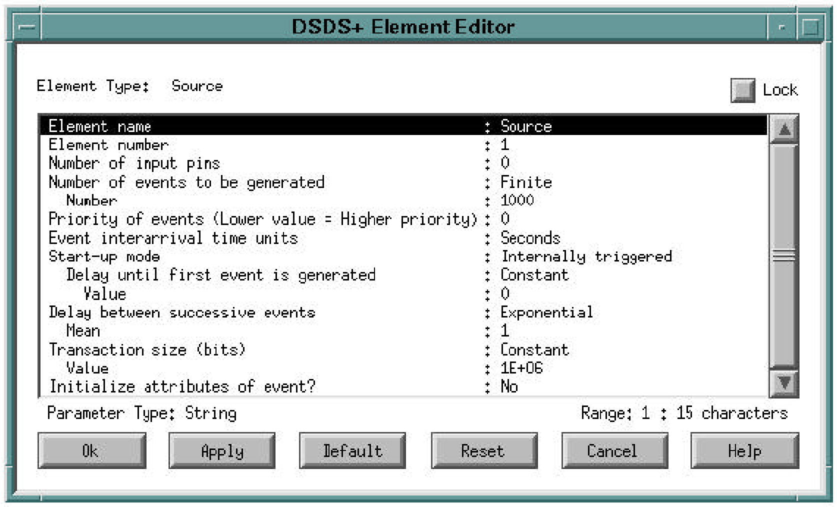
Simulating a Model

After selecting a model and scenario, selecting the "Simulate" button will start execution of the simulation as a background process. A message will be displayed in the transcript showing the id assigned to the process by the operating system. When the process ends, this number will be used to report its termination status.

After selecting a model and scenario and executing the simulation, the "Reports" button will activate the Report Browser. The Report Browser is a text file browser which provides access to the various DSDS+ reports. Reports include simulation initialization, runtime messages, event summary, point-to-point statistics, element statistics, queue size statistics, queue entry statistics and simulation trace. The Plot Editor is for viewing plots of DSDS+ timeline data. Types of plots include storage, utilization, throughput and queue statistics. Plots can also be customized by the user. Figure 5 shows an example plot and report.

OPTIMIZATION TECHNIQUES IN DSDS+

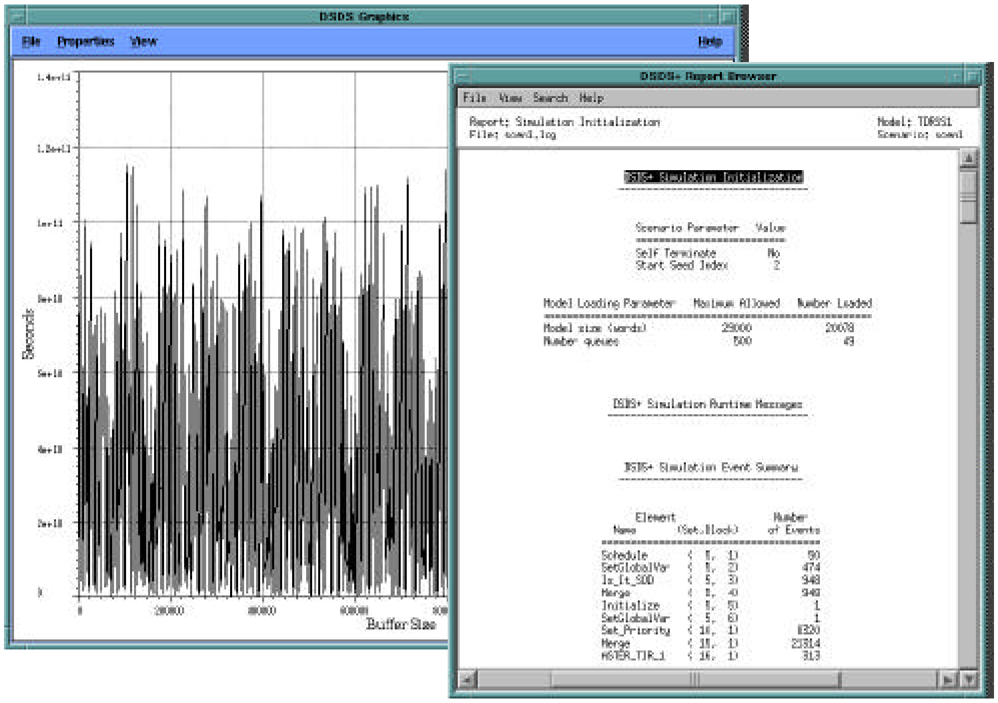
Large complex data systems create an extremely large volume of data. A limiting factor when modeling these data systems is the amount of simulation (CPU) run time required to complete the simulation. The CPU run time



```
Element Type:   Source

Element name                                      : Source
Element number                                    : 1
Number of input pins                              : 0
Number of events to be generated                  : Finite
  Number                                          : 1000
Priority of events (Lower value = Higher priority): 0
Event interarrival time units                     : Seconds
Start-up mode                                     : Internally triggered
  Delay until first event is generated            : Constant
    Value                                         : 0
Delay between successive events                   : Exponential
  Mean                                            : 1
Transaction size (bits)                           : Constant
  Value                                           : 1E+06
Initialize attributes of event?                   : No
Parameter Type: String                      Range: 1 : 15 characters
```

Figure 4: Element Editor with Source element parameters.



```
Report: Simulation Initialization                     Model: TDRSS1
File: scen1.log                                       Scenario: scen1

              DSDS+ Simulation Initialization
              -------------------------------

                Scenario Parameter    Value
                =========================
                Self Terminate         No
                Start Seed Index        2

   Model Loading Parameter   Maximum Allowed   Number Loaded
   ==========================================================
   Model size (words)             25000            20678
   Number queues                    500               49


              DSDS+ Simulation Runtime Messages
              --------------------------------


              DSDS+ Simulation Event Summary
              ------------------------------

              Element                  Number
          Name        (Set,Block)    of Events
          ====================================
          Schedule       (  5,  1)         50
          SetGlobalVar   (  5,  2)        474
          Is_It_SOD      (  5,  3)        948
          Merge          (  5,  4)        948
          Initialize     (  5,  5)          1
          SetGlobalVar   (  5,  6)          1
          Set_Priority   ( 16,  1)       8320
          Merge          ( 15,  1)      23514
          ASTER_TIR_1    ( 16,  1)        313
```

Figure 5: Report and plot browsers.

increases when the number of events in the simulation increases. During a simulation, each packet generated creates an event. As the packets flow through the model (being queued, processed, sorted, etc.) more events are created, thus increasing the CPU run time of the simulation.

For large complex data systems with extremely large volumes of data the number of events generated can become overwhelming. Since the purpose of doing a simulation is to try different configurations, determine potential bottlenecks, and test out the effects of different components with different costs and performance, it is important to be able to run a simulation many times in a timely fashion. This allows different versions of the model to be compared to determine which components would work best given system constraints, such as price and performance.

To bring simulation run times down to a reasonable value, modelers often use optimization techniques. The next two sections describe two different optimization techniques and their use with DSDS+.

Artificially-Inflated Packets

When modeling a system, the data that passes through it is divided into packets. Packets (or messages) may be characterized by size. In standard DSDS+, packet size is a parameter that is supplied by the user. The size of these data packets can vary, but the typical NASA packet size is in the range of 10,000 bits. Very large numbers of packets are associated with the simulation of current and emerging NASA data systems. Every packet transfer results in a state change within the model. Each state change requires CPU run time. For instance, to simulate a 24 hour "true" EDOS model it would have taken more than a day of CPU time.

As complexity, data rates and need for longer run times increases, so does the need for an optimization technique. Artificially raising the packet size is one way of optimizing the simulation. By increasing the packet size, say from 10,000 to 100,000 bits, CPU run time is reduced by a factor of 10 due to the decrease in events flowing through the system. Unfortunately, increasing the size of packets results in errors. The magnitude of the error can be predicted by comparing the results to a "truth" model. A truth model is one constructed with the "real" packet size. It is compared to a test model which is constructed with the elevated packet size. However, it is impractical to develop a truth model for some large systems, such as EDOS which when fully configured, would consist of more than 100 payloads (experiments). Before reaching users, payload data must pass through 10 or more processing points. Thus it is not easy to determine the effects on the modeling results when the packets are inflated artificially.

Data Streams

The data streams technique is another way of optimizing the modeling process. As stated above, the objective is to reduce the number of events in a given simulation run. The data streams method models rate changes rather than individual packets. Consider the example of an experiment which transmits a 10,000 bit packet at the constant rate of 100 packets per second. Consider also that the experiment transmits 10 minutes each orbit. In a "true" model this translates into 600,000 packets (events) per 10 minute duty cycle. A data stream represents this duty cycle as two events; namely one start and one end. The data stream would be characterized as a 10 minute stream with a transmission rate of 600,000 bits per second.

The key to understanding the data stream methodology is that it takes advantage of the linear flow of data between state changes. The speed at which the simulation runs for the data streams method is not dependent on the volume of packets, but instead is dependent on the number of times the data rate changes. The data streams method requires less computation and thus reduces the time to simulate the passing of data through the system. Data streams take advantage of the fact that data systems behave linearly between state changes. Therefore, data streams can model the effects of the changes in the data rates of a system rather than modeling each individual packet. This optimizing method also reduces the CPU run time of a simulation due to the decrease in events.

One difference between packet and data stream modeling is the way a processor's bandwidth is allocated. A packet, no matter its size, occupies the entire bandwidth of its processor for some finite period of time. Data streams, on the other hand, occupy only that portion of the bandwidth which is equal to or less than its transmission rate. Furthermore, data streams share the bandwidth proportionally on a first in, first out basis with other competing streams.

Comparison of Data Streams to Packet Modeling

Data stream simulation is tantamount to modeling with infinitesimally small (approximately 1 bit) packets. In terms of magnitude, 1 bit is closer to 10,000 than 20,000 bits and more. Based upon this empirical point alone, an assumption could be made that errors induced by data stream optimization would be less.

An experiment was performed to assess the error produced by artificially increasing the packet size of a packet model of a system and the error introduced by data streams modeling. Models using these optimization techniques were compared to a "Truth Model" - a model which is run using the actual packet size. Figure 6 shows the run times of the Truth Model where the packet sizes were 15 Kilobits (reflecting the actual implementation), a model where the packet size was artificially expanded to 1 Megabit packets, and the data streams methodology. As expected, the Truth Model ran the longest, 3,279 CPU seconds. The expanded packet model ran for 47 seconds and the data streams model for 62 seconds.

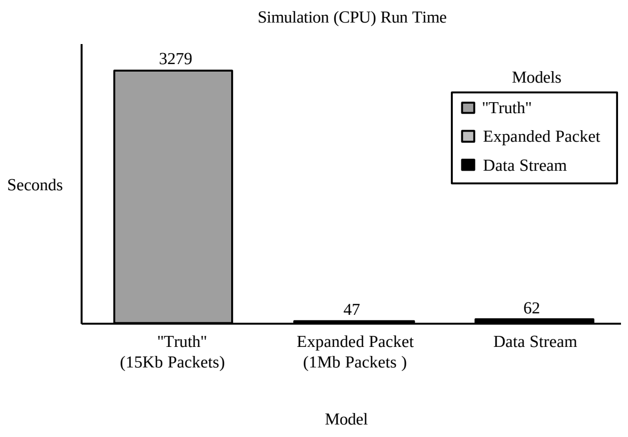


Figure 6: Run times of "truth", expanded packet and data stream models.

Figure 7 illustrates the percentage of difference for the data's mean transit times from source to sink of the expanded packet model and the data stream model when compared to the truth model. As can be seen, there is a significant amount of error introduced in the mean transit times of the data when the packet size is artificially increased. The mean transit time errors in the data stream model were negligible when compared to the expanded packet model for all six payloads.

5. CONCLUSION

DSDS+ has been used on many projects at NASA to evaluate candidate architectures for end-to-end data systems. It provides a direct manipulation graphical user interface and hierarchical model development that provides close correlation between the representation and real system. The combination of the packet modeling, data streams modeling and extensibility makes the tool versatile and suitable for many different modeling situations. The data streams optimization technique allows accurate modeling of high data rate systems that could not be modeled accurately using the expanded packet optimization technique. As more high data rate systems are used in government and commercial applications, a technique such as data streams will be come essential for modeling their performance in a timely and accurately manner.

DSDS+ has been used by Johnson, Marshall and Goddard Space Flight Centers. Projects that have used DSDS+ include ESDIS (GSFC), Space Station (at JSC), and Advanced Xray Astronomical Facility (MSFC). DSDS+ is available on Sun and DEC workstations running under UNIX and the X Window System.

DSDS+ and documentation can be obtained by anonymous ftp from kong.gsfc.nasa.gov in /pub/dsds. The DSDS+ home page is at URL http://groucho.gsfc.nasa.gov/Code_520/Code_522/Projects/DSDSPlus/.

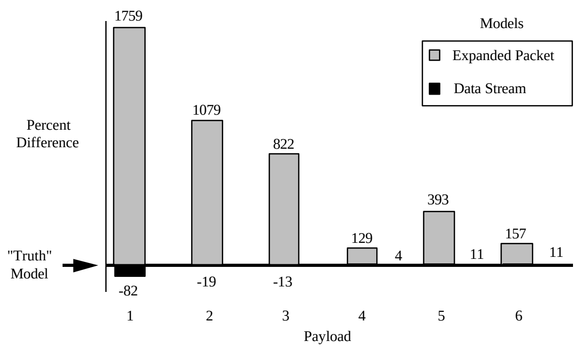


Figure 7: Mean transit time difference.

6. REFERENCES

[1] Data Systems Dynamic Simulator (DSDS+) User's Guide. Version 1.2. June 1996. Data Systems Technology Division Report number DSTL-96-001.
[2] Scheifler, R. and Gettys, J. 1986. The X Window System. ACM Transactions on Graphics. April. 79-109.
[3] Linton, M. A., Vlissides, J. M. and Calder, P. R. 1989. Composing User Interfaces with Interviews. IEEE Computer, February, 1989.
[4] Data Systems Dynamic Simulator (DSDS+) Standard Library Manual. Version 1.2. June 1996. Data Systems Technology Division Report number DSTL-96-002.